APPROXIMATE EDGE 3-COLORING OF CUBIC GRAPHS

A Thesis
Presented to
The Academic Faculty

by

Amita S. Gajewar

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Computing

Georgia Institute of Technology
August 2008

APPROXIMATE EDGE 3-COLORING OF CUBIC GRAPHS

Approved by:

Professor Richard Lipton, Advisor
College of Computing
Georgia Institute of Technology

Professor Dana Randall
College of Computing
Georgia Institute of Technology

Professor H. Venkateswaran
College of computing
Georgia Institute of Technology

Date Approved: 26 June 2008

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor, Dr. Richard Lipton, for the encouragement and guidance for this thesis work. While he gave me a freedom to work on different problems of my interest, he also nudged me in the proper direction to ensure that I stay focused. I would also like to thank Atish Das Sarma, Subrahmanyam Kalyanasundaram and Danupon Nanongkai for the guidance and motivation. Without the continuing support by Dr. Lipton and my collegues this thesis would not have been possible to realize. I would also like to thank other thesis committe members, Dr. Dana Randall and Dr. H. Venkateswaran, for their precious time and support.

TABLE OF CONTENTS

SUMMARY

The work in this thesis can be divided into two different parts. In the first part, we suggest an approximate edge 3-coloring polynomial time algorithm for cubic graphs. For any cubic graph with n vertices, using this coloring algorithm, we get an edge 3-coloring with at most $\frac{n}{3}$ error vertices. In the second part, we study Jim Propp's Rotor-Router model on some non-bipartite graph. We find the difference between the number of chips at vertices after performing a walk on this graph using Propp model and the expected number of chips after a random walk. It is known that for line of integers and a grid Z^d, this deviation is constant. However, it is also proved that for k-ary infinite trees, for some initial configuration the deviation is no longer a constant and say it is D. We present a similar study on some non-bipartite graph constructed from k-ary infinite trees and conclude that for this graph with the same initial configuration, the deviation is almost k^2D.

CHAPTER I

APPROXIMATE EDGE 3-COLORING OF CUBIC GRAPHS

1.1 Introduction

For any cubic graph, finding a valid edge 3-coloring is an NP-hard problem. Therefore, it will require an exponential time to compute one. However, we can use a polynomial time algorithm which will give an approximate edge 3-coloring. We present such alogorithm, say A, to edge 3-color any cubic graph. We claim that for a cubic graph with n vertices, the resulting edge 3-coloring yields at most $\frac{n}{3}$ error vertices. To validate this claim, we prove that this coloring of G, has at least two unique correct vertices for every error vertex. In order to map a correct vertex to at most one error vertex we use a mapping approach, say B. Sections 1.3 and 1.4 describe the algorithm A and the mapping approach B, respectively. Sections 1.5, 1.6 and 1.7 provide the proofs to obtain the main result.

1.1.1 Definitions

Definition An *edge 3-coloring* of a graph G is a mapping of edges of G to colors $\in \{1, 2, 3\}$. A *valid* edge 3-coloring is an edge 3-coloring such that no two edges incident on a common vertex map to the same color.

Definition v is an *error-vertex* in an edge 3-coloring of a cubic graph G if two edges incident on v map to the same color. The number of errors in a coloring of G is the number of error-vertices.

1.2 Previous Work

According to Vizing's theorem, any graph with maximum degree k can be edge-colored correctly with $(k+1)$ colors. Thus, a cubic graph can be edge-colored correctly with 4 colors because in this graph every vertex has a degree 3. Since a cubic graph with n vertices will have $\frac{3n}{2}$ edges, in the correct edge-coloring of this graph, there exists some color which is used for at most $\frac{3n}{8}$ edges. If we color these edges with one of the remaining three colors then at most $\frac{3n}{8}$ vertices will have adjacent edges with the same color. Therefore, according to Vizing's theorem, there exists an edge 3-coloring of a cubic graph with n vertices such that we get at most $\frac{3n}{8}$ error vertices.

1.3 Coloring Algorithm A

Let G be any cubic graph and v be any vertex with v_1, v_2 and v_3 as adjacent vertices. Assume v_1 is adjacent to m and n, v_2 is adjacent to o and p, and v_3 is adjacent to s and t respectively.

1. Run BFT(Breadth First Traversal) on G and let T be the result of it such that G is now represented by T in an hierarchical form, with a root vertex at the level 0. Let d be the depth of T. Let $l = 0$

2. If $l < d$ then $l = l + 1$ else go to step 4.

3. Let v be any vertex at level l such that it is not yet colored. Let v_1, v_2 and v_3 be the adjacent vertices of v. Assign colors to edges incident to v so as to color it correctly without changing already correct vertices to error vertices. If v can be colored correctly then proceed to step e. If not, then go to step a.

   (a) Check if any vertex at a distance 1 from v is an error vertex. If such vertex exists, call it z. If assigning a different color to $e(v, z)$ makes v correct then make the necessary changes and proceed to step e. If v cannot be colored

correctly or there is not an error vertex at a distance 1, then proceed to step b.

(b) Check if any vertex at a distance 2 from v is an error vertex. If yes, call it z. Now, z and v must be having a common neighbor, either v_1, v_2 or v_3. Without loss of generality, assume that z is adjacent to v_1. Now, exchange the colors of edges $e(v1, z)$ and $e(v1, v)$. If this makes v correct without making v_1 incorrect, then continue to step e. If not, then mantain the previous coloring and go to step c.

(c) Check if any vertex at a distance 2 is at level $(l + 1)$. If yes, call it z. As explained in (b), let z be adjacent to v_1. Then exchange the colors of edges $e(v_1, z)$ and $e(v_1, v)$. If this makes v correct without making v_1 incorrect, proceed to step e. If not, then mantain the previous coloring and go to step d.

(d) If there exists more than one error vertices within a distance 2 from v, then try the above color exchange scheme in combination for all these vertices. If v is still an error vertex, then keep the original coloring and proceed to step e.

(e) Repeat step 3 to color the next vertex at the level l. If all the vertices at level l are colored then goto step 2.

4. Assign $l = 0$

5. If $l < d$ then $l = l + 1$ else go to step 8.

6. Let x be the correct vertex at level l with two non-parent adjacent vertices y and z (vertices at level l or $(l + 1)$). If both y and z are not error vertices then go to step 7. Let u be the parent node of x. If u has a non-parent adjacent node (other than x), call it w. If not, go to step 7.

(a) If w is not an error vertex, continue to step (b). Otherwise, exchange the color of edges $e(x, u)$ and $e(w, u)$.

(b) If w is correct and it also has two non-parent error adjacent vertices then exchange the color of edges $e(x, u)$ and $e(w, u)$.

Note: If the condition in (a) or (b) is satisfied, then changing the color of $e(u, x)$ might make x an error vertex. But, since both y and z are adjacent error vertices, we can always color x correctly by asigning appropriate color to edges $e(x, y)$ and/or $e(x, z)$. This change might infact make it possible to color y and/or z correctly. Further, note that in (a), since w is already an error vertex changing the color of the edge $e(u, w)$ will not create any new error vertex. In case of (b), since w's adjacent vertices are error vertices, it is possible to keep w correct even after the color of the edge $e(u, w)$ is changed.

7. Repeat step 6 for the next correct vertex at the level l. If all correct vertices at level l are examined by step 6, then goto step 5.

8. End

1.4 Vertex Mapping Approach B

This section describes an approach to map a correct vertex to *at most* one error vertex. Let x be the correct vertex at level l.

1. x has one non-parent adjacent vertex, say y

   (a) y is an error vertex

   Map x to y.

   (b) y is a correct vertex

   If y has two non-parent adjacent error vertices, say u_1 and u_2 then map y to u_1 and map x to u_2. Otherwise, leave x unmapped.

2. x has two non-parent adjacent vertices, say, y and z

   (a) Either y or z is an error vertex

   Map x to the error vertex.

   (b) Both y and z are error vertices

   Map x to either y or z.

   (c) If both y and z are correct vertices

   Map x to one of the non-parent adjacent vertices of y and z if need be. However, in order to ensure that it is mapped to only one error vertex we impose the condtion: Either y or z can have two non-parent adjacent vertices as error vertices. Say, only z has two non-parent adjacent error vertices, u_1 and u_2. Then, map x to u_1 and z to u_2.

3. x has no non-parent adjacent node

   Leave x unmapped.

1.5 Main Result

Theorem 1.5.1. *Let G be a cubic graph with n vertices. If it is colored using the above mentioned coloring algorithm A, then it has at most $\frac{n}{3}$ error vertices.*

Proof. In order to prove that G has at most $\frac{n}{3}$ error vertices, we need to show that for every error vertex, there exist two unique correct vertices and that every correct vertex is mapped to at most one error vertex. The approach B used to map a correct vertex to error vertex ensures that a correct vertex is mapped to at most one error vertex. Now, we need to prove that for every error vertex there exist two correct vertices. According to *claim* 1.7.1, v can have at most one adjacent error vertex. Without loss of generality let's assume, v_1, v_2 are correct and v_3 may or may not be a correct vertex. Since, v is an error vertex, at least two of its adjacent edges should be of the same color. Let's assume $c(v_1, v) = c(v_2, v)$.

1. v_3 is an error vertex

   According to *claim* 1.7.1, only one of the v_1, v_2, v_3 can be an error vertex. Here v_3 is an error vertex. According to the second part of the *claim* 1.7.1, v_1 and v_2 can have only v as the adjacent error vertex. Therefore, v_1 and v_2 can be mapped to v.

2. v_3 is correct and v_1 has three adjacent error vertcies

   According to *claim* 1.7.4, only one of the v_1, v_2, v_3 can have all of its neighbors as error vertices. Here, it is v_1. Therefore, according to *claim* 1.7.4, v_2 and v_3 can have only v as the adjacent error vertex. Therefore, v_2 and v_3 can be mapped to v.

3. v_1, v_2 and v_3 are all correct and have at most two non-parent adjacent error vertices (including v)

   According to *claim* 1.6.3, there exist two correct vertices within a distance 2 from v which can be mapped to v using B.

This proves the main result. □

1.6 Claims

For all the claims henceforth, assume that v is any error vertex in the graph colored using the algorithm A on G. As described before, v has v_1, v_2 and v_3 as adjacent vertices. And v_1 is adjacent to m and n, v_2 is adjacent to o and p, and v_3 is adjacent to s and t. Further, since v is an error vertex, its two or more edges are colored correctly. Without loss of generality we will assume that $c(v_1, v) = c(v_2, v)$. And $c(v_3, v)$ may or may not be equal to $c(v_1, v)$. Further, unless otherwise stated, assume that v_1, v_2 and v_3 are not adjacent to each other and are colored correctly.

Claim 1.6.1. *Assume $c(v, v_1) = c(v, v_2) \neq c(v, v_3)$. If v has an error vertex at a distance 2, say m, such that m is adjacent to v_1 (or v_2) then $c(v_3, v) = c(v_1, m)$ (or*

$c(v_3, v) = c(v_2, m)$*).*

Proof. Assume $c(v_3, v) \neq c(v_1, m)$. Since v_1 is correct, $c(v_1, m) \neq c(v_1, v) = c(v_2, v)$. Now, according to step $3(b)$ of A, by exchanging the color $c(v_1, m) \leftrightarrow c(v_1, v)$, we get $c(v_1, v) \neq c(v_3, v) \neq c(v_2, v)$. This makes v a correct vertex which is a contradiction. Therefore, $c(v_3, v) = c(v_1, m)$. □

Claim 1.6.2. *Assume $c(v, v_1) = c(v, v_2) = c(v, v_3)$. If v has two (or three) error vertices at a distance 2, say m and p (and s) such that they are adjacent to v_1 and v_2 (and v_3) respectively, then $c(v_1, m) = c(v_2, p)$ $(= c(v_3, s))$.*

Proof. Assume $c(v_1, m) \neq c(v_2, p)$. According to step $3(b)$ of A, by exchanging the color, $c(v_1, m) \leftrightarrow c(v_1, v)$ and $c(v_2, p) \leftrightarrow c(v_2, v)$ we get $c(v_1, v) \neq c(v_3, v) \neq c(v_2, v)$. This results in v being correct vertex which is a contradiction. Therefore, $c(v_1, m) = c(v_2, p)$.

Similarly, we can prove that $c(v_1, m) = c(v_3, s)$.

$\Rightarrow c(v_1, m) = c(v_2, p) = c(v_3, s)$. □

Claim 1.6.3. *Assume v_1, v_2 and v_3 have at most two adjacent error vertices (including v). Then, for every such v there exist two correct vertices within a distance 2 which can be mapped to v using B.*

Proof. We shall divide the proof in two parts depending upon the whether or not v_1, v_2 and v_3 are adjacent to each other.

Part 1 - v_1, v_2 and v_3 are not adjacent to each other .

Part 2 - v_1 is adjacent to v_2; v_3 may or may not be adjacent to v_1 and/or v_2.

Proof for Part 1 - v_1, v_2 and v_3 are not adjacent to each other According to the approach B defined above, v_1, v_2 and v_3 can be associated with only one error vertex. If any two of the v_1, v_2 and v_3 have only v as the non-parent adjacent error vertex

then we are done according to step 2(a) of B. Assume that, at least two of these three correct vertices, have two non-parent adjacent error vertices (including v). Therefore, according to *claim* 1.7.3, v_1, v_2 and v_3 cannot have all neighbors in common. Let's consider two main cases:

1. v_1 and v_2 have 2 common neighbors: $m = p$ and v
2. v_1 and v_2 have 1 common neighbor: v

1. v_1 and v_2 have 2 common neighbors: $m = p$ and v

   (a) $m = p$ is a parent node

   Since $m = p$ is a correct node, $c(v_1, m = p) \neq c(v_2, m = p)$. We know that $c(v_1, v) = c(v_2, v)$. Therefore, $c(v_1, n) \neq c(v_2, o)$. Therefore, according to *claims* 1.6.1 and 1.6.2, both n and o cannot be error vertices. Let's say n is an error vertex and o is correct vertex. Then, according to 2(c) in B, $m = p$ can be mapped to v and v_1 to n. Further, since o is correct vertex, we can map v_2 to v. Therefore, we get two correct vertices for $v : m = p$ and v_2 that are within a distance 2.

   (b) $m = p$ is a child node

   Since v_1 and v_2 should have at least two non-parent error neighbors, assume $m = p$ is also an error vertex. According to *claim* 1.7.2, in this case it is possible to satisfy the condition stated in step 2(c) of B and we can map n and o to $m = p$ (or v) and leave v_1 and v_2 to be mapped to v (or $m = p$). (Note: since $m = p$ is a child node, n and o are parent vertices of v_1 and v_2 respectively).

2. v_1 and v_2 have 1 common neighbor: v

   As per our assumption, at least two of the v_1, v_2, v_3 have two non-parent adjacent error vertices, including v. If v_1 and v_2 have two non-parent adjacent error

vertices, then according to *claim* 1.7.2, and as explained above in *case* 1(b), we can map n and o to m and p respectively, and leave v_1 and v_2 to be mapped to v. (Note: n and o are parent vertices of v_1 and v_2 respectively). In addtion, if s is also an error vertex then, v_3 can be mapped to it becasue, v_1 and v_2 are already mapped to v.

Proof for Part 2 - v_1 is adjacent to v_2; v_3 may or may not be adjacent to v_1 and/or v_2.

1. v_1 and v_2 have different parent vertices

   In this case, both v_1 and v_2 have only v as a child node and therefore they can be mapped to v as mentioned in 1(a) step of B.

   (Note: Further, if v_3 is adjacent to v_1 or v_2 then $v_3 = m$ or $v_3 = p$. However, even then v_1 and v_2 can be associated to v).

2. v_1 and v_2 have common parent node, say $m = p$

   This case cannot arise where $c(v_1, v) = c(v_2, v)$. However, v can be an error vertex such that $c(v_3, v) = c(v_1, v)$ or $c(v_3, v) = c(v_2, v)$. In any case, both v_1 and v_2 have only v as a child node and therefore they can be mapped to v as mentioned in 1(a) step of B.

   (Note: Further, if v_3 is adjacent to v_1 then, $v_3 = m = p$. Even then v_1 and v_2 can be mapped to v).

3. v_1 is a parent node of v_2

   (a) v_3 is adjacent to v_1

   v_1 is a parent node of v_2 and is adjacent to v. Therefore, if v_3 is adjacent to v_1 then it will be a parent node of v_1. However, v_3 is also adjacent to v in which case v will have to be at the same level as that of v_1. And then v_2 becomes a child of v. Therefore, v will be colored before v_2 and so it will

be colored correctly, however, v_2 may or may not be. So, this case cannot arise.

(b) v_3 is adjacent to v_2

Then $v_3 = p$. And our assumption is that v_1, v_2 and v_3 are correct vertices. Now, $v_3 = p$ is a correct vertex and so v_1, v_2 have v as the only non-parent adjacent error vertex. Therefore, we can map v_1 and v_2 to v.

(c) p is correct and v_3 is not adjacent to v_1 and v_2

If p is a correct vertex then v_1, v_2 have v as the only non-parent adjacent error vertex. Therefore, we can map v_1 and v_2 to v.

(d) p is an error vertex and v_3 is not adjacent to v_1 and v_2

Let s be v_3's non-parent adjacent vertex. If s is a correct vertex, then v_1 and v_3 have v as the only non-parent adjacent error vertex and so we can map v_1 and v_3 to v. Now, assume that s is an error vertex. Then, according to *claim* 1.7.2, v_2's parent node v_1 cannot have an non-parent error adjacent vertex. However, v is its non-parent error adjacent vertex which is a contradiction. However, v_1 is a parent node of v_2 and therefore, v is a sibling So, this case cannot arise.

□

1.7 Supporting Claims

Claim 1.7.1. *v can have at most one error vertex at a distance* 1*, say* v_1*. And if* v_1 *is an error vertex then, v cannot have any error vertex at a distance* 2*.*

Proof. If v_2 or v_3 is an error vertex in addition to v_1, then we can color the edges $e(v_1, v)$ and $e(v_2/v_3, v)$ appropriately so that v is colored correctly. But, v is an error vertex and therefore, v_2 and v_3 should be correct vertices. Further, since v is an error vertex it is required that $c(v_2, v) = c(v_3, v)$. Assume that v_2's adjacent vertex, say p,

is an error vertex. Since v_2 is a correct vertex, $c(v_2, v) \neq c(v_2, p)$. So, if we exchange the color of these edges then we get $c(v_2, v) \neq c(v_3, v)$. Further, v_1 is an error vertex and hence we can achieve $c(v_2, v) \neq c(v_3, v) \neq c(v_1, v)$. This makes v a correct vertex. However, v is an error vertex and therefore, v_2 and v_3 cannot have any adjacent error vertex (except v). □

Claim 1.7.2. *If v_1 and v_2 have two non-parent error adjacent vertices (say n, v and o, v respectively), then v_1 and v_2 should have correct parent vertices, say u_1 and u_2 respectively. Further, u_1 and u_2 can have v_1 and v_2 as the only non-parent error adjacent vertex. And, u_1's and u_2's other adjacent vertex (other than v_1, v_2) cannot have both non-parent adjacent vertices as error vertices.*

Proof. Let m and p be the parent vertices of v_1 and v_2. According to *claims* 1.6.1 and 1.6.2, since n and o are error vertices, $c(v_1, n) = c(v_2, o)$. And as mentioned before, $c(v_1, v) = c(v_2, v)$. This results in $c(v_1, m) = c(v_2, p)$. We shall consider the following two cases:

1. $m = p$

   Since, $c(v_1, m) = c(v_2, p)$, $m = p$ is an error vertex. However, this cannot happen because an algorithm A colors one level at a time and so $m = p$ will be colored before v_1 and v_2, and since $m = p$ has only one parent node and two child nodes, only one of its edges will have color determined beforehand. Therefore, it can always be colored correctly. Therefore, this case cannot arise.

2. $m \neq p$

   (a) m or p is an error vertex

   If m is an error vertex, then v_1 will have all error neighbors and according to *claim* 1.7.4, v_2 cannot have any adjacent error vertex. However, v_2's

adjacent vertex, o, is an error vertex. This is a contradiction. Therefore, m cannot be an error vertex. Similarly we can prove that p cannot be an error vertex. Therefore, this case cannot arise.

(b) m and p are correct

($b1$) $u_1,\ u_2$ have non-parent error adjacent vertices, say s_1, s_2 respectively.

Then, according to step $6(a)$ in algorithm A, we can exchange the colors $c(m, v_1) \leftrightarrow c(m, s_1)$ such that $c(v_1, m) \neq c(v_2, p)$. Thus, making $c(v_1, n) \neq c(v_2, o)$ which is a contradiction (Since n and v are error vertices, after changing $c(v_1, m), v_1$ can still be correct by coloring $e(v_1, v)$ or $e(v_1, n)$ appropriately). Therefore, s_1 should be correct and similarly we can prove that s_2 should be correct.

($b2$) s_1 is correct and has two error non-parent adjacent vertices.

Then again as in ($b1$), following the step $6(b)$ in A, we can change the color $c(m, v_1)$ which results in $c(v_1, m) \neq c(v_2, p)$. This contradicts the *claims* 1.6.1 and 1.6.2. Therefore, v_1's sibling cannot have two error non-parent adjacent vertices.

This proves the claim. □

Claim 1.7.3. *If v_1 and v_2 have all their neighbors in common, say $m = p, n = o$ and v, then $v_1, v_2, m = p$ and $n = o$ should be correct.*

Proof. Either $m = p$ or $n = o$ should be a parent node of v_1 and v_2. Let's say, $m = p$ is a parent node. Since $m = p$ is a parent node with v_1 and v_2 as child vertices, it cannot be an error vertex. Therefore, $c(v_1, m = p) \neq c(v_2, m = p)$. We know, $c(v_1, v) = c(v_2, v)$. Therefore, $c(v_1, n = o) \neq c(v_2, n = o)$. According to *claim* 1.6.1 and 1.6.2, if $n = o$ is an error vertex, $c(v_1, n = o) = c(v_2, n = o)$. This leads to a contradiction. Thus, $v_1, v_2, m = p$ and $n = o$ need to be correct. □

Claim 1.7.4. *Assume v has no error vertex at a distance 1. Say, v_1's all adjacent vertices are error vertices then v_2, v_3 cannot have any adjacent error vertex.*

Proof. Assume v_2's adjacent vertex, say p, is an error vertex. Then, according to *claim* 1.6.1 and 1.6.2, $c(v_1, m) = c(v_2, p) = c(v_1, n)$. This leads to a contradiction because it makes v_1 an error vertex. Therefore, p cannot be an error vertex. Therefore, v_2 and v_3 cannot have adjacent error vertex, except v. □

1.8 Conclusion and Future Work

We suggest an approximate edge 3-coloring polynomial time algorithm for cubic graphs. And we prove that for any cubic graph with n vertices, using this coloring algorithm, we get an edge 3-coloring with at most $\frac{n}{3}$ error vertices. We believe that it can be a very good substitute for applications where a random edge 3-coloring is used. This substitution will be good for two main reasons. First reason is that our suggested bound on the number of error vertices is better than the expected error vertcies in random edge 3-coloring. And the second reason is that this coloring algorithm is also a polytime algorithm and therefore fast enough.

In future, we intend to extend this work in order to achieve a bound of at most $\frac{n}{4}$ errors (or may be better).

CHAPTER II

DETERMINISTIC RANDOM WALKS USING ROTOR-ROUTER MODEL

2.1 Introduction

Jim Propp's rotor-router model is a simple deterministic process first introduced by Priezzhev et al. [6] and later popularized by Jim Propp. It can be viewed as an attempt to de-randomize random walks on graphs. In Propp model, each vertex x is equipped with a 'rotor' together with a cyclic permutation (called a 'rotor sequence') of the cardinal directions corresponding to the edges incident to the vertex. While a chip (particle, coin, . . .) performing a random walk leaves a vertex in a random direction, in the Propp model it always goes in the direction the rotor is pointing. After a chip is sent, the rotor is rotated according to the fixed rotor sequence. This rule ensures that chips are distributed quite evenly among the neighbors of a vertex. The Propp machine has attracted considerable attention recently. It has been shown that it closely resembles a random walk in several respects. In [5], it has been shown that, if an (almost) arbitrary number of chips are placed on the vertices of $\mathbb{Z}^d$-grid, and a walk using a Propp model is performed, then the number of chips deviates from the expected number of chips obtained from random walk, by at most a constant.
This result raises the question if all graphs do have this property. In [1], a rotor-router model was studied to find the discrepancy between the random walk and the walk performed using rotor-router model on k-regular infinite trees. It answers the forementioned question negatively by proving that for any deviation D, there is an initial configuration of chips such that after running the Propp model for certain time, there is a vertex with at least D more chips than expected in the random walk model.

Here, we study of rotor-router model on a non-bipartite graph obtained from the k-regular infinite trees. The *section* 2.2 provides more insight into the construction of this graph. We study this non-bipartite graph for the same initial configuration as that for k-regular inifinite trees in [1] and the result is that we get a single-vertex discrepancy of at least $(k-1)(k-2)D$ because of the non-bipartite nature as opposed to D in case of k-regular infinite trees. The *sections* 2.4, 2.5 and 2.6 provide the proofs for this main result.

2.2 Our Sample Graph

The rotor-router model has been studied primarily on bipartite graphs and for initial even or odd configurations, i. e., chip configurations supported on vertices at an even or odd distance from the origin. In order to study the same model on non-bipartite graph, we start with a very simple graph that is obtained from k-regular infinite tree. Let's call this graph as G. Now, we will describe the construction of this graph. If k is even, add $k/2$ edges to connect the k nodes at the level 1 such that every node is connected to exactly one other node at level 1. If k is odd, then add $\frac{k-1}{2}$ edges to connect the $(k-1)$ nodes leaving the k^{th} node without any cross edge. In order to maintain the k-regular property of G, all these nodes at level 1 with one cross edge, will have $(k-2)$ children at level 2 and if a node does not have a cross edge it will have $(k-1)$ children. All the nodes at level 2 or more will have $(k-1)$ children. Owing to the non-bipartite nature of the graph, the chips on the odd and even vertices of G will interfere with each other.

2.3 Preliminaries

Let $G = (V, E)$ be the graph constructed from k-regular infinite trees, as described in the previous section. We fix a root node to be its origin 0. $\mid x \mid$ denotes the shortest distance between the origin and vertex x. A configuration describes the current state of the Propp machine. A configuration of the Propp machine assigns to each

vertex $x \in V$ its current (integral) number of chips and the current direction of the rotor. For all $x \in V$ and $t \in \mathbb{N}$, let $f(x,t)$ denote the number of chips on vertex x and $ARR(x,t)$ the direction of the rotor associated with x after t steps of the Propp machine. In other words, $f(\cdot,t)$ is the configuration function at time t. We will use $x + ARR(x,t)$ to denote the node at which the current rotor of x is pointing at time t.

A vertex discrepancy at time t, is the difference in the number of chips present at time t on that vertex using a Propp model and the expected number of chips present after performing a random walk for t steps. In order to find the single vertex discrepancy between Propp machine and random walk on G, we can treat the expectation of the random walk as a linear machine [3]. To describe the linear machine we use the same fixed initial configuration as for the Propp machine. In one step, each vertex x sends a $1/k$ fraction of its (possibly fractional) number of chips to each neighbor. Let $E(x,t)$ denote the number of chips at vertex x after t steps of the linear machine. This is equal to the expected number of chips at vertex x after a random walk of all chips for t steps. Let $DIR = \{-1,\ 0,\ +1\}$. Note that by definition

$E(x,t) = \frac{1}{k} \sum_{A \in DIR} E(x + A, t-1)$

A random walk on G can be described by its probability density. Let $H(x,t)$ denotes the probability that a chip from the vertex at a distance x to the root vertex arrives at the root vertex after t random steps in a simple random walk. Then,
$H(x,t) = \frac{n(x,t)}{k^t}$

where $n(x,t)$ counts the number of t-length paths between origin and vertex at a distance x in G. It is easy to verify the following properties of $n(x,t)$:
$n(0,0) = 1$

$n(x,0) = 0$ for all $x \geq 1$

$n(0,t) = kn(1,t-1)$ for all $t \geq 1$

$n(1,t) = n(0,t-1) + n(1,t-1) + (k-2)n(2,t-1)$ for all $t \geq 1$

$n(x,t) = n(x-1,t-1) + (k-1)n(x+1,t-1)$ for all $x \geq 2, t \geq 1$

Finally, we write $x \sim t$ to mean that $\mid x \mid \equiv t(mod\ 2)$.

2.4 Mod-k forcing Theorem

Now, since Propp machine is a deterministic process, it is obvious that the initial configuration (the location of each chip and the direction of each rotor) determines all subsequent configurations. However, following theorem, shows a partial converse, that (roughly speaking) we may prescribe the number of chips *modulo k* on all vertices at all times by finding an appropriate initial configuration. It can easily be proved by induction. An analogous result for the one-dimensional Propp machine has been shown in [2] and for k-regular infinite trees is shown in [1].

Theorem 2.4.1. *(Mod-k forcing Theorem) For any initial direction of the rotors and any $\pi : V \times \mathbb{N} \to 0, 1, ..., (k-1)$, there is an initial configuration $f(x,0)$ that results in subsequent configurations satisfying $f(x,t) \equiv \pi(x,t)(mod\ k)$ for all x and $t \geq 0$.*

Proof. Analogous to the proofs given in [2] and [1], we start with $f(x,0) = \pi(x,0)$ chips at location x. Now, assume that our initial configuration is such that for some $T \in \mathbb{N}$, we have $f(x,t) \equiv \pi(x,t)(mod\ k)$ for all $t < T$. We modify this initial configuration by defining $f\prime(x,0) = f(x,0) + \epsilon_x k^T$. Here, $\epsilon_x \in \{0, 1, ..., (k-1)\}$ are to be determined such that $f\prime(x,t) \equiv \pi(x,t)(mod\ k)$ for all $t \leq T$. Observe that, addition of a pile of k^T chips splits those chips evenly among neighboring nodes for all $t < T$ times. Therefore, for all choices of ϵ_x, we have $f\prime(x,t) \equiv \pi(x,t)(mod\ k)$ for all $t < T$. Let $\epsilon_x = 0$ for all x with $\mid x \mid < T$. This implies that we modify the number

of chips in the initial configuration only for the vertices $x \in V$, such that $\mid x \mid \geq T$. By induction on $\mid x \mid$, we fix the ϵ_x such that $f\prime(x,T) \equiv \pi(x,T)(mod\ k)$ for all x. Assume that for some $\theta \in \mathbb{N}$, the current ϵ_x fulfill $f(x,T) \equiv \pi(x,T)(mod\ k)$ for all x with $\mid x \mid < \theta$. For all x with $\mid x \mid = \theta$, we choose some y with $\mid y \mid = T+\theta$ such that the (shortest) distance between x and y is T and set $\epsilon_y = Abs((\pi(x,T) - f\prime(x,T)))mod\ k$. For all other y with $\mid y \mid = T+\theta$, we set $\epsilon_y = 0$.

This yields the existence of $\epsilon_y, y \in V$, such that $f\prime(x,t) \equiv \pi(x,t)(mod\ k)$ for all $t = T$ and $x \in V$. Further, at any given position x, the initial number of chips will be constant after the first $\mid x \mid$ iterations. This means that the process converges to the sought-after initial configuration. However, note that, nodes at level 1 are connected to some other nodes at level 1. Therefore, for every x, $f(x,t)$ may have non-zero value. In the proofs mentioned in papers [2] and [1], if $x \not\sim t$, then $f(x,t) = 0$ because of the bipartite nature of the graph. However, since the graph G is not a bipartite, this condition may not hold. Therefore, as described above, in order to ensure that $f\prime(x,T) \equiv \pi(x,T)(mod\ k)$, we will be choosing ϵ_y and y even if $\mid y \mid$ is odd. □

2.5 Basic Method

Here, we summarize the idea mentioned in [1] to analyze the maximal possible single-vertex discrepancy. Let $E(x,t_1,t_2)$ denote the number of chips at location x, after first performing t_1 steps with the Propp machine and then $(t_2 - t_1)$ steps with the linear machine. $\mid f(x,t) - E(x,t) \mid$ denotes the discrepancy for all vertices x and all times t. In order to study the divergence of the Propp model and the linear machine, let's consider the vertex $x = 0$. From definition,

$E(0,0,t) = E(0,t)$.

$E(0,t,t) = f(0,t)$.

This implies,

$f(0,t) - E(0,t) = \sum_{s=0}^{t-1}(E(0,s+1,t) - E(0,s,t))$

with,

$E(0,s+1,t) - E(0,s,t)$

$= \sum_{x \in V} \sum_{l=1}^{f(x,s)}(H(|\ x\ +\ NEXT^{l-1}(ARR(x,s))|, t-s-1) - H(|\ x\ |, t-s))$.

where $NEXT$ denotes the next position of the rotor.

From this we obtain the definition of the Influence of a Propp move (compared to a random walk move) from vertex x in direction $A \in \{-1, 0, +1\}$ on the discrepancy of 0 (t time steps later) as:

$INF(x, A, t) = H(x + A, t-1) - H(x, t)$

In order to ultimately reduce all ARRs involved to the initial arrow settings $ARR(\cdot, 0)$, we define

$s_i(x) = \min\{u \geq 0 \mid i < \sum_{t=0}^{u} f(x,t)\}$ for all $i \in \mathbb{N}$.

Hence at time $s_i(x)$ the location x is occupied by its i-th chip.

Consider the discrepancy at 0 at time T . Then the above yields

$f(0,T) - E(0,T)$

$= \sum_{x \in V} \sum_{i \geq 0, s_i(x) < T} INF(|\ x\ |, NEXT^i(ARR(x,0)), T - s_i(x))$.

Since the inner sum of equation will occur frequently in the remainder, let us define the contribution of a vertex x to be

$con(x) = \sum_{i \geq 0, s_i(x) < T} INF(|\ x\ |, NEXT^i(ARR(x,0)), T - s_i(x))$

where we both suppress the initial configuration leading to the $s_i(\cdot)$ as well as the runtime T. This lead to the following theorem.

Theorem 2.5.1. *The discrepancy between Propp machine and linear machine after T time steps is the sum of the contributions $con(x)$ of all vertices x, i. e.,*

$f(0,T) - E(0,T) = \sum_{x \in V} con(x)$

2.6 Divergence of the models

In this section, we analyze a specific initial configuration and show that the Propp machine may deviate from the linear machine by an arbitrarily large number of chips. For a fixed time T at which we aim to maximize the discrepancy $f(0,T) - E(0,T)$ we examine a configuration in which all vertices x with $0 <\mid x \mid \leq \frac{T}{\lambda}$ and $\lambda = \frac{k}{k-2}$ are occupied by a number of chips not divisible by k only once. We assume that at that time $T - t_{|x|}$ with $t_x = \lceil \lambda x \rceil$ a chip is sent in the direction of 0. Such a configuration exists by *Theorem* 1. We will prove the following theorem.

Theorem 2.6.1. *For any initial direction of the rotors and any $T > 0$, there is an initial configuration such that the single vertex discrepancy between the Propp machine and linear machine after T time steps is $\Omega((k-1)(k-2)\sqrt{kT})$.*

Proof. 1. k is even.

$f(0,T) - E(0,T)$

$= \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

$= \sum_{x=1}^{\frac{T}{\lambda}} k((k-1)^{x-1} - (k-1)^{x-2})(H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

$= \sum_{x=1}^{\frac{T}{\lambda}} k((k-1)^{x-1} - (k-1)^{x-2})(\frac{n(x-1,t_x-1)}{k^{t_x-1}} - \frac{n(x,t_x)}{k^{t_x}})$

$= \sum_{x=1}^{\frac{T}{\lambda}} \frac{(k-1)^{x-1} - (k-1)^{x-2}}{k^{t_x-1}} (kn(x-1, t_x - 1) - n(x, t_x))$

$= \sum_{x=1}^{\frac{T}{\lambda}} \frac{(k-1)^{x-1} - (k-1)^{x-2}}{k^{t_x-1}} i(x, t_x)$

with $i(x,t) = kn(x-1, t-1) - n(x,t)$ *for all* $x, t \geq 1$

In order to evaluate this equation, let's obtain $i(x,t)$ in a recursive form like

$i(x,t) = i(x-1, t-1) + (k-1)i(x+1, t-1)$

Let us define $i(0,t) = i(x,0) = 0$.

$i(1,1) = k - 1$

For $x = 1$ and $t \geq 2$ we have

$i(1,t) = kn(0,t-1) - n(1,t)$

$= (k-1)n(0,t-1) - (k-2)n(2,t-1) - n(1,t-1)$

$= (k-1)(kn(1,t-2) - n(2,t-1)) + n(2,t-1) - n(1,t-1)$

$= i(0,t-1) + (k-1)i(2,t-1) + n(2,t-1) - n(1,t-1)$

For $x = 2$ and $t \geq 2$ we have

$i(2,t) = kn(1,t-1) - n(2,t)$

$= kn(0,t-2) + kn(1,t-2) + k(k-2)n(2,t-2) - n(1,t-1) - (k-1)n(3,t-1)$

$= i(1,t-1) + (k-1)i(3,t-1) + kn(1,t-2) - kn(2,t-2)$

$= i(1,t-1) + (k-1)i(3,t-1) + k(n(1,t-2) - n(2,t-2))$

For $x \geq 2$ and $t = 1$ we have

$i(x,1) = kn(x-1,0) - n(x,1)$

$= 0$

$= i(x-1,0) + (k-1)i(x+1,0)$

For $x > 2$ and $t > 2$ we get

$i(x,t) = kn(x-1,t-1) - n(x,t)$

$= kn(x-2,t-2) + k(k-1)n(x,t-2) - n(x-1,t-1) - (k-1)n(x+1,t-1)$

$= i(x-1,t-1) + (k-1)i(x+1,t-1)$

Now that we have obtained

$i(x,t) = i(x-1,t-1) + (k-1)i(x+1,t-1)$ for all $x,t > 2$, we need to show that

$i(2,t) > i(1,t-1) + (k-1)i(3,t-1)$

For that we prove the following claim

Claim 2.6.2. *$n(1,t) > n(2,t)$ for all $t > 2$*

Proof. As defined, $n(2,t)$ denotes the number of paths starting from node at

level 2 and ending at root node(a node at level 0) in t *steps*. We will show that for every such path, say p, there exists at least two unique ways to construct a path, say q from node at level 1 to node at level 0 in t *steps*. Therefore, $n(1,t) > n(2,t)$. Let w, y and z denote the nodes at level $0, 1$ and 2 respectively. We know that p starts from z and ends at w. Let's construct a path q such that, $q = e(y,z) + p$. Thus, q is a path from y to w and takes $t+1$ steps. Further let $p = q\prime + e(u,y\prime) + e(y\prime,w)$ where $y\prime$ is a node at level 1 and it may or may not be equal to y and u is a node at level 1 or 2. Consider two main cases:

(a) p's first edge is $e(z,y)$

We can rewrite p as, $p = e(z,y) + q\prime\prime + e(u,y\prime) + e(y\prime,w)$ where $q\prime = e(z,y) + q\prime\prime$. Now, set $q = q\prime\prime + e(u,y\prime) + e(y\prime,w)$. Then, q will be a path of $t-1$ steps.

i. u is a node at level 2

Expand q such that $q = q\prime\prime + e(u,y\prime) + e(y\prime,v) + e(v,w)$ where v is the vertex at level 1 and is adjacent to $y\prime$.

Alternatively, if $q\prime\prime$ has its first edge as $e(y,v_1)$ where v_1 is a node at level 1 and is adjacent to y, then, we can set $q = e(y,w) + e(w,v_1) + q\prime\prime\prime + e(u,y\prime) + e(y\prime,w)$ where $q\prime\prime = e(y,v_1) + q\prime\prime\prime$. And if $q\prime\prime$ has its first edge as $e(y,w)$ then we can set $q = e(y,v_1) + e(v_1,w) + q\prime\prime\prime + e(u,y\prime) + e(y\prime,w)$ where $q\prime\prime = e(y,w) + q\prime\prime\prime$

ii. u is a node at level 1

Expand q such that $q = q\prime + e(u,y\prime) + e(y\prime,u) + e(u,w)$ or $q = q\prime + e(u,z\prime) + e(z\prime,u) + e(u,w)$ where $z\prime$ is the vertex at level 2 and is adjacent to u.

(b) p's first edge is not $e(z,y)$

We know that $p = q\prime + e(u, y\prime) + e(y\prime, w)$. Now set $q = e(y, z) + q\prime + e(u, y\prime) + e(y\prime, w)$. Therefore, q will be a path of $t + 1$ steps.

i. u is a node at level 1

   In this case, set $q = e(y, z) + q\prime + e(u, w)$. Now, q has only t steps.

   Alternatively, we can reduce the length of cycle $q\prime$ by 2, call it $q\prime\prime$. And set $q = e(y, z) + q\prime\prime + e(u, y\prime) + e(y\prime, u) + e(u, w)$.

ii. u is a node at level 2

   Let $q\prime = q\prime\prime + e(z, y) + q\prime\prime\prime$. $q\prime\prime$ is a cycle which starts from z and comes back to z, in say, q steps. We can shorten the cycle $q\prime\prime$ such that it has $(q - 2)$ steps, say $r\prime$. This will make path q of steps $(t - 1)$. We can modify q to have exactly t steps. This can be done by setting $q = e(y, z) + r\prime + e(z, y) + q\prime\prime\prime + e(u, y\prime) + e(y\prime, y\prime\prime) + e(y\prime\prime, w)$. Further, depending upon the edges that are removed to shorten the cycle $q\prime$, we get different paths.

Therefore, (1) and (2) implies that we get at least two unique ways to construct a path q of t steps from y to w for each path p from z to w. □

Summarizing the above, we see that $i(x, t)$ can be defined recursively as follows

$i(x, 0) = 0$ for all x

$i(0, t) = 0$ for all t

$i(1, 1) = k - 1$

Let $z = n(1, t - 1) - n(2, t - 1)$

$i(1, t) = i(0, t - 1) + (k - 1)i(2, t - 1) - z$

$i(2, t) = i(1, t - 1) + (k - 1)i(3, t - 1) + kz$

$i(x, t) = i(x - 1, t - 1) + (k - 1)i(x + 1, t - 1)$ for $(x, t) \geq (3, 3)$

As described in [1], this recursive view of $i(x,t)$ reveals another interpretation of these quantities. Apart from a factor $(k-1)^{(\frac{t-x}{2}+1)}$, $i(x,t)$ counts the number of lattice paths from the origin $(0,0)$ to (x,t) of steps $(+1,+1)$ and $(-1,+1)$which do not cross the line $x=0$. This can be described by the well-known Ballot numbers. Further, z can be at least 1 and thus $i(2,t)$ adds additional value at least $(k-1)$ to every such lattice path. Thus,

$i(x,t) > (k-1)^{(\frac{t-x}{2}+2)} \frac{x}{t} \binom{t}{\frac{t+x}{2}}$

Therefore,

$f(0,T) - E(0,T) > \sum_{x=1}^{T/\lambda} \frac{((k-1)^{x-1} - (k-1)^{x-2})}{k^{tx-1}} (k-1)^{(\frac{t_x-x}{2}+2)} \frac{x}{t} \binom{t}{\frac{t+x}{2}}$

$= \sum_{x=1}^{T/\lambda} \frac{((k-1)^{\frac{t_x+x}{2}+1} - (k-1)^{(\frac{t_x+x}{2})}}{k^{tx-1}} \frac{x}{t} \binom{t}{\frac{t+x}{2}}$

$= \sum_{x=1}^{T/\lambda} \frac{((k-1)^{(\frac{\lambda+1}{2}x+1)} - (k-1)^{(\frac{\lambda+1}{2}x)}}{\lambda k^{\lambda x-1}} \binom{t}{\frac{t+x}{2}}$

$= \sum_{x=1}^{T/\lambda} \frac{(k-1)^{(\frac{\lambda+1}{2}x)}(k-1-1)}{\lambda k^{\lambda x-1}} \binom{t}{\frac{t+x}{2}}$

$= \sum_{x=1}^{T/\lambda} \frac{(k-1)^{(\frac{k-1}{k-2}x)}(k-2)^2}{k k^{\frac{k}{k-2}x-1}} \binom{t}{\frac{t+x}{2}}$

$= \sum_{x=1}^{T/\lambda} \frac{(k-1)^{(\frac{k-1}{k-2}x)}(k-2)^2}{k^{\frac{k}{k-2}x}} \binom{t}{\frac{t+x}{2}}$

Using Stirling formula we obtain,

$\binom{n}{k} > \frac{n^{\frac{n+1}{2}}}{3(n-k)^{n-k+1/2} k^{(k+1)/2}}$

$\Rightarrow \binom{t}{\frac{t+x}{2}} > \frac{k^{\frac{k}{k-2}x+\frac{1}{2}}(k-2)^{\frac{1}{2}}}{3(k-1)^{\frac{k-1}{k-2}x+\frac{1}{2}}\sqrt{x}}$

$\Rightarrow f(0,T) - E(0,T) > k(k-2) \sum_{x=1}^{T/\lambda} \frac{(k-2)^{\frac{3}{2}}}{6k^{\frac{1}{2}}(k-1)^{\frac{1}{2}}\sqrt{x}}$

Similar to the main result (Theorem 3) in [1],

$= \Omega(k(k-2)\sqrt{kT})$

2. k is odd

Here, the root node has one child node at level 1 with no cross edge and it has k child nodes as in k-regular tree. Therefore,

$f(0,T) - E(0,T)$

$= \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

$= \sum_{x=1}^{\frac{T}{\lambda}} (k-1)((k-1)^{x-1} - (k-1)^{x-2})(H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|})) +$

$((1)(k-1)^{x-1})(H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

The part 1 of the above sum can be calculated similar to the proof mentioned above. Therefore,

$\sum_{x=1}^{\frac{T}{\lambda}} (k-1)((k-1)^{x-1} - (k-1)^{x-2})(H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

$> (k-1)(k-2) \sum_{x=1}^{T/\lambda} \frac{(k-2)^{\frac{3}{2}}}{6k^{\frac{1}{2}}(k-1)^{\frac{1}{2}}\sqrt{x}}$

$= \Omega((k-1)(k-2)\sqrt{kT})$

The part 2 of the above sum can be calculated from the main result (Theorem 3) in [1],which evaluates as follows,

$\sum_{x=1}^{\frac{T}{\lambda}} ((1)(k-1)^{x-1})(H(\mid x \mid -1, t_{|x|} - 1) - H(\mid x \mid, t_{|x|}))$

$= \Omega(\frac{1}{k}\sqrt{kT})$

Therefore, for $k > 1$

$f(0,T) - E(0,T) = \Omega((k-1)(k-2)\sqrt{kT})$

From part 1 and 2, we can obtain the result with $k > 1$,

$f(0,T) - E(0,T) = \Omega((k-1)(k-2)\sqrt{kT})$

So far in the proof, we considered the single vertex discrepancy only at root vertex(origin). Let's consider the case for a vertex, say y such that $\mid y \mid = 1$. Therefore,
$f(1,T) - E(1,T) = \frac{1}{k} \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H_1(\mid x \mid -1, t_{|x|} - 1) - H_1(\mid x \mid, t_{|x|}))$

where $H_1(x,t)$ denotes the probability that a chip at a distance x to the origin arrives to y in t random steps.
$H_1(x,t) = \frac{n_1(x,t)}{k^t}$
where $n_1(x,t)$ denotes the number of $t-$length paths to y from a vertex at a distance $\mid x \mid$ to root. It can be rewritten in the form of $n(x,t)$ as
$n_1(x,t) = n(x, t+1)$
Therefore, $H_1(x,t) = \frac{n_1(x,t)}{k^t} = \frac{n(x,t+1)}{k^t} = kH(x,t+1)$

$f(1,T) - E(1,T)$
$= k\frac{1}{k} \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H(\mid x \mid -1, t_{|x|}) - H(\mid x \mid, t_{|x|} + 1))$

After solving this equation as described above,
$= \sum_{x=1}^{T/\lambda} \frac{(k-1)^{(\frac{k-1}{k-2}x + \frac{1}{2})}(k-2)^2}{k^{\frac{k}{k-2}x}} \binom{t+1}{\frac{t+x+1}{2}}$
From Lemma 4 in [1], $\binom{2x}{x+y}$ is monotonic nondecreasing function in x for $0 \leq y \leq x$.Therefore,
$\binom{t+1}{\frac{t+x+1}{2}} \geq \binom{t}{\frac{t+x}{2}}$
$\Rightarrow f(1,T) - E(1,T) > \sum_{x=1}^{T/\lambda} \frac{(k-1)^{(\frac{k-1}{k-2}x + \frac{1}{2})}(k-2)^2}{k^{\frac{k}{k-2}x}} \binom{t}{\frac{t+x}{2}}$

This can be solved in a similar way as shown in the proof.
Likewise, let $H_2(x,t)$ denote the probability that a chip at a distance x to the origin arrives to, say y, such that $\mid y \mid = 2$, in t random steps.
$H_2(x,t) = \frac{n_2(x,t)}{k^t}$

where $n_2(x,t)$ denotes the number of $t-$length paths to y from a vertex at a distance $\mid x \mid$ to root. It can be rewritten in the form of $n_1(x,t)$ as

$n_2(x,t) = \frac{k-1}{k} n_1(x,t+1)$

Therefore, $H_2(x,t) = \frac{k-1}{k}\frac{n_2(x,t)}{k^t} = \frac{k-1}{k}\frac{n_1(x,t+1)}{k^t} = \frac{k-1}{k} k H_1(x,t+1) = \frac{k-1}{k} k^2 H(x,t+2)$

We can generalize this as,

$H_y(x,t) = \frac{k-1}{k} k^y H(x,t+y)$

where $H_y(x,t)$ denotes the number of $t-$length paths to a vertex at a distance $\mid y \mid$ from a vertex at a distance $\mid x \mid$ to root. Now,

$f(y,T) - E(y,T)$

$= \frac{1}{k(k-1)^{y-1}} \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H_y(\mid x \mid -1, t_{|x|} - 1) - H_y(\mid x \mid, t_{|x|}))$

$= k^y \frac{k-1}{k} \frac{1}{k(k-1)^{y-1}} \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H(\mid x \mid -1, t_{|x|} + y - 1) - H(\mid x \mid, t_{|x|} + y))$

$> \sum_{x \in V, |x| \leq \frac{T}{\lambda}} (H(\mid x \mid, t_{|x|} + y - 1) - H(\mid x \mid, t_{|x|} + y))$

This can be solved as explained above and therefore, we can conculde the divergence of the models to be $\Omega((k-1)(k-2)\sqrt{kT})$. □

2.7 Conclusion And Future Work

We study Jim Propp's Rotor-Router model on some non-bipartite graph whose construction is described in *section* 2.2. We conclude that for this graph, there exists some initial configuration such that the single vertex discrepancy is at least $\Omega((k-1)(k-2)\sqrt{kT})$. This result shows that the deviation between the walk performed by Propp model and a random walk is not constant for this graph.

In future, we would like extend this work for more generic graphs that are non-bipartite. In particular, it will be interesting to establish the relation between the possible deviation and the number of cycles in the graph.

REFERENCES

[1] Cooper, J., Doerr, B., Friedrich, T., and Spencer, J., "Deterministic random walks on regular trees," *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pp. 766–772, 2008.

[2] Cooper, J., Doerr, B., Spencer, J., and Tardos, G., "Deterministic random walks on the integers," *European Journal of Combinatorics*, pp. 2072–2090, 2007.

[3] Cooper, J. and Spencer, J., "Simulating a random walk with constant error," *Combinatorics, Probability and Computing*, 2006.

[4] Diestel, R., *Graph Theory.* Third Edition, Springer-Verlag, Heidelberg Graduate Texts in Mathematics, 1997.

[5] Doerr, B. and Friedrich, T., "Deterministic random walks on the two dimensional grid," *arXiv:math*, 2007.

[6] Priezzhev, V. B., Dhar, D., Dhar, A., and Krishnamurthy, S., "Eulerian walkers as a model of selforganized criticality," *Physical Review Letters*, 1996.

[7] T. Hoffman, J. M. and Schmeichel, E., "On edge-coloring graphs," *Combinatorica*, pp. 119–128, 1992.

[8] Tait, P. G., "Note on a theorem in geometry of position," *Trans. Roy. Soc. Edinburgh*, pp. 657–660, 1880.

[9] Tutte, W. T., "On hamiltonian circuits," *J. London Math. Soc.*, pp. 98–101, 1946.